Distinguishing Attacks on Stream Ciphers Based on Arrays of Pseudo-random Words*

Nathan Keller† and Stephen D. Miller‡

Abstract

In numerous modern stream ciphers, the internal state consists of a large array of pseudo-random words, and the output key-stream is a relatively simple function of the state. In [16], it was heuristically shown that in various cases this structure may lead to distinguishing attacks on the cipher. In this paper we further investigate this structural attack. We present a rigorous proof of the main probabilistic claim used in the attack in the basic cases, and demonstrate by examining a concrete example (the cipher SN3 [11]) that the heuristic assumptions of the attack are remarkably precise in more complicated cases. Furthermore, we use the general technique to devise a distinguishing attack on the stream cipher MV3 [9] requiring 2^{82} words of key-stream. Unlike the attacks in [16], our attack does not concentrate on the least significant bits of the words, thus allowing to handle the combination of more operations (XORs, modular additions and multiplications, and rotations by a fixed number of bits) in the update and output rules of the cipher.

*This is the full version of a paper submitted for publication in a journal, which contains only Sections 1, 2, and 3. The material in Section 4 concerns an attack on the MV3 stream cipher. After writing up a description of our results, we learned that essentially identical arguments – but with important miscalculations – had simultaneously been published in [15]. For the sake of completeness we include an appendix reconciling the two attacks. See footnote 2 for similar comments on the SN3 stream cipher.

†This author is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

‡Partially supported by NSF grant DMS-0601009.

1 Introduction

Stream ciphers are widely used in practical cryptography, especially in environments where the high speed of encryption is crucial. While in general there exist many different types of stream ciphers, there are several structures which are shared by numerous modern ciphers.

One of these structures is basing the internal state of the cipher on a very large array, where the output key-stream is a relatively simple function of the array elements. The first cipher having this structure was alleged RC4 (designed in 1987), and modern ciphers of this class include HC-256 [19], the Py family [2], SN3 [11], NGG [13], GGHN [8], MV3 [9], and others (see [16]). The main advantage of ciphers of this class is their high speed: due to the simple update and output rules, the encryption is very fast, while the security follows from pseudo-random accesses to a very large array, unknown to the attacker.

In alleged RC4, the array consists of a permutation of the values $0, 1, \ldots, 255$. In the modern ciphers, which are based on longer (usually, 32-bit) words, keeping a permutation of all the possible words in memory is infeasible, and hence the permutation is replaced by an array of pseudo-random words.

In [16], Paul and Preneel showed heuristically that in various cases, the *randomness* of the array elements can be used to mount a distinguishing attack on the cipher.[1] Their method is based on the following phenomenon:

Proposition 1. *Let T be an array of n independently and uniformly chosen random values in $\{0,1\}$, and let i, j be chosen independently and uniformly at random from $\{1,\ldots,n\}$. Then*

$$Pr\,[\,T[i] = T[j]\,] \;\; = \;\; \frac{1}{2} + \frac{1}{2n}.$$

This calculation comes from the fact that if $i = j$ then $T[i] = T[j]$ holds for sure, and if $i \neq j$, then by randomness, $Pr[T[i] = T[j]] = 1/2$. Put another way:

> the XOR of randomly chosen bits from a fixed, random array is *not* uniformly distributed, but has a slight bias towards zero.

[1] We note that the practical significance of distinguishing attacks of the class presented in [16], as well as those discussed in our paper (attacks that are unlikely to be leveraged to a key-recovery attack), can be questioned (see [18]). This issue is outside of the scope of the current paper.

This bias potentially applies to any cryptographic operation summing random entries from random arrays. Note that the attack does not work if the values of the array are not random (like in alleged RC4, where at any stage of the encryption, exactly half of the LSBs of the array elements are zeros). This is because in the case $i \neq j$, there is a small *counter bias*, which makes the overall probability unbiased. The main idea of the attack in [16] (originally presented in [7]) is to find a condition E on the elements of the array, such that if E is satisfied, then the least significant bit (LSB) of a combination of the output words is biased; this bias is used to distinguish the output of the cipher from a random string.

In this paper we further investigate the structural attack presented in [16]. First, we present a rigorous generalization of Proposition 1 to compute the bias of the sum of k random elements, for all $k \geq 1$. Then we demonstrate the precision of the heuristic assumption (i.e., the assumption that if the special event E does not occur, then there is no counter-bias) in more complicated cases by considering a concrete example. We devise a distinguishing attack on the stream cipher SN3 [11] designed by Maltchev in 2002. Our heuristic computation shows that the bias of the distinguisher is $2^{-15.40}$, and a series of experiments with a key-stream of length 2^{40} shows that the bias is in the range between $2^{-15.37}$ and $2^{-15.42}$. Hence, the heuristic assumption is remarkably precise in this case.[2]

Finally, we use the general technique to mount a distinguishing attack on MV3. The stream cipher MV3 [9] was designed by Keller *et al.* in 2006, as an attempt to generalize the alleged RC4 structure to a 32-bit word based cipher. The internal state of the cipher consists of several large buffers of pseudo-random words, and the update and output rules are based on rapidly mixing random walks. The cipher uses a combination of different arithmetic operations on 32-bit words (XORs, additions, and multiplications), as well as cyclical rotations by a fixed number of bits. While the encryption speed of MV3 is high (less than 5 cycles per byte), the initialization phase is very slow, and hence the cipher is appropriate for encrypting large amounts of data. The security claim made in [9] is that no attack on the cipher should be faster than exhaustive key search for 256-bit keys. As mentioned in the

[2]We note that another distinguishing attack on SN3 was presented in [14]. The attack uses part of the observations used in our attack, along with different techniques. The authors of [14] claim that the attack requires $2^{28.2}$ words of key-stream. However, a careful examination of the attack, presented in the appendix, shows that it requires about 2^{38} words of key-stream, while our attack requires less than 2^{30} words.

footnote on the first page, this attack was independently codiscovered in [15], but with a mistaken complexity estimate. No other cryptographic attacks on the cipher have been presented thus far.

Our distinguishing attack on MV3 requires 2^{82} words of key-stream. The attack uses the higher-order differential technique [10], examining the XOR of quadruples of output words having a special structure. The main feature of the attack is that it does not concentrate on the least significant bits of the output words. Instead, the attack considers the XOR of pairs of consecutive bits of the output words, following a technique presented by Cho and Pieprzyk [5] in a different context. As a result, the attack can handle fixed bit rotations and modular multiplications, unlike previous attacks of the same structure (e.g., [16, 19]) that can handle only rotations by a *pseudo-random* number of bits, modular additions and XORs.[3] While our distinguishing attack is clearly impractical, it is faster than the security claims asserted in [9].

The paper is organized as follows: in Section 2 we present the proof of the generalization of Proposition 1. The attack on SN3, along with the experimental verification of the results, is presented in Section 3. In Section 4 we present the attack on MV3. In the appendix we discuss the flaws in the attacks presented in [14] and [15].

2 Rigorous Proof of the Heuristic Assumption in Basic Cases

In this section we present a rigorous generalization of Proposition 1 to the sum of k random 0/1 elements, for all $k \geq 1$. We give both an exact formula and a simpler asymptotic for large arrays.

Proposition 2. *Let T be an array of N independently distributed uniform random values in $\{0, 1\}$, and let $k \geq 1$. Let S_k denote the sum of k elements of the array which are chosen uniformly and independently at random (including*

[3] This partially settles two open problems raised in ([17], Section 7.2), asking for a generalization of the techniques presented in [16] which is not restricted to analyzing the LSB, and which can handle also modular multiplications.

possible repetitions). We have that

$$p_k \;:=\; Pr[\,S_k \equiv 0 \pmod 2\,] \;=\; \frac{1}{2} + \frac{1}{2}\sum_{r=0}^{N}\left[2^{-N}\frac{N!}{(N-r)!r!}\right]\left(\frac{2r}{N}-1\right)^k;$$

in particular $p_k = \frac{1}{2}$ *for* k *odd,* $p_2 = \frac{1}{2} + \frac{1}{2N}$*, and* $p_4 = \frac{1}{2} + \frac{3}{2N^2} - \frac{1}{N^3}$*. For* N *large,* p_{2k} *is approximately* $\frac{1}{2} + \frac{1}{2}(\frac{2}{N})^k \pi^{-1/2}\Gamma(k+1/2)$*, where* Γ *denotes the Gamma function.*

Proof. Let $t_1, \ldots, t_N$ denote the elements of the array, and consider the random variable

$$X \;:=\; \sum_{i=1}^{N}(-1)^{t_i} \;=\; 2Y - N\,,$$

where $Y = \sum_{i \le N} t_i$ is the $Binomial(N, 1/2)$ random variable whose probability distribution is given by the bracketed expression above. Expanding out this sum for k-th powers of X, we see

$$X^k \;=\; \sum_{i_1,\ldots,i_k=1}^{N}(-1)^{t_{i_1}+\cdots+t_{i_N}} \;=\; 2\,\#\{1 \le i_1, \ldots, i_k \le N \,|\, t_{i_1}+\cdots+t_{i_k} \equiv 0 \pmod 2\} - N^k.$$

Combining these formulas shows that p_k is determined by the expected value of moments of Y:

$$p_k \;=\; \frac{\#\{1 \le i_1, \ldots, i_k \le N \,|\, t_{i_1} + \cdots + t_{i_k} \equiv 0\}}{N^k} \;=\; E\left[\frac{1}{2}(\frac{2Y}{N} - 1)^k\right] + \frac{1}{2}.$$

This establishes the formula asserted above, because this expected value is its second term. The special cases of p_k mentioned can be seen by direct calculation.

As far as the limit of p_{2k} for N large, we recall the Central Limit Theorem: that the $Binomial(N, 1/2)$ random variable is approximated by the Normal distribution with mean $N/2$ and variance $N/4$ (see [6, Chapter VII]). This allows us to approximate the sum above with the integral

$$\int_{\mathbb{R}}\left[\sqrt{\frac{2}{\pi N}}e^{-\frac{(2x-N)^2}{2N}}\right]\left(\frac{2x}{N}-1\right)^{2k} dx\,,$$

which — after changing variables to $y = \sqrt{2x - N}$ — may be computed as $(\frac{2}{N})^k \pi^{-1/2}\Gamma(k+1/2)$ using Euler's integral formula $\Gamma(s) = \int_0^\infty e^{-u}u^{s-1}du$. □

We use Proposition 2 as the basic tool in many of our probability calculations in what follows. In order to derive estimates using it, we need to make a working assumption that the array elements and pointers in the stream ciphers we study are themselves independently, identically uniformly distributed at random. Of course the point of this paper is to contradict that, by demonstrating an attackable bias. However, it seems numerically reasonable to make this assumption as an approximation, because a deviation that is strong enough to significantly affect our calculations would itself likely produce a stronger attack than the ones we present. In addition, in the complicated cases where the bias cannot be proved rigorously, we follow [16] in making the heuristic assumption that there is no counter-bias when the special event does not occur. The experiment presented in the next section demonstrates the precision of this assumption.

3 Distinguishing Attack on SN3

3.1 Description of SN3

The stream cipher SN3 [11] was designed by Simeon Maltchev in 2002. SN3 is a software-efficient stream cipher, optimized for execution on 32-bit micro processors. It is based on a large array of pseudo-random words, and uses only simple word operations, like XOR and cyclical rotations.

The structure of SN3 is the following: The internal state of the cipher consists of three arrays $V1$, $V2$, and $V3$, of sixty four 32-bit words each, and three 6-bit indices, i, j, and m. Index i addresses only the $V1$ array, index j addresses only the $V2$ array, and index m addresses only the $V3$ array.

The key-stream generation is composed of 64-step cycles, where in each cycle, i takes the values from 0 to 63 sequentially, and j and m perform a (pseudo)-random walk, determined by the elements of the $V1$ array. In each step, one word from each array is selected according to i, j, and m, the XOR of these three words is outputted as the key-stream word, and the three words are used to update themselves by a relatively simple update rule.

The structure of a step is outlined in the pseudo-code below. In the code, $\oplus$ denotes bitwise XOR, $\lll$ denotes cyclical left rotation, and $\gg$ denotes noncyclical right shift (which discards the rightmost bits). The key-stream word which is outputted at the end of the step is denoted K_i. The indices i and j are set to zero at the beginning of the key-stream generation process.

1. $T1 = V1[i]$
2. $T2 = V2[j]$
3. $m = T1 \pmod{64}$
4. $T3 = V3[m]$
5. $V1[i] = (T1 \lll 1) \oplus T2$
6. $V2[j] = (T2 \lll 5) \oplus T3 \oplus 0x8c591ca1$
7. $V3[m] = (T3 \lll 17) \oplus T1 \oplus 0xab8ec254$
8. $i = i + 1 \pmod{64}$
9. $j = T1 \gg 8 \pmod{64}$
10. $K_i = T1 \oplus T2 \oplus T3$

After each 64-step cycle, the arrays are rotated cyclically to the left: the contents of $V1$ are placed in $V3$; those of $V3$, in $V2$; and those of $V2$, in $V1$. We omit the key expansion algorithm, since it does not affect our attacks. Instead, we assume that the initial values in the arrays $V1$, $V2$, and $V3$ are independently uniformly distributed.

3.2 Our Attack

The basic observation we use in the attack is a weakness in the update rule of $V1, V2$, and $V3$. We observe that regardless of the values of the words $T1$, $T2$, and $T3$ before the update operation, the values $V1[i]$, $V2[j]$, and $V3[m]$ after the update satisfy a simple linear relation involving the Hamming weight (the number of 1's in a word's binary representation). If these words are next updated simultaneously, this will result in a bias that is described below.

Proposition 3. *Consider the values $V1[i]$, $V2[j]$, and $V3[m]$ right after the update operation in step 7 above. The parity of the Hamming weight of $V1[i] \oplus V2[j] \oplus V3[m]$ is zero, that is, its binary representation has an even number of 1's.*

Proof. Let $g(x)$ denote the parity of the Hamming weight of a 32-bit word x, i.e., $g(x) \equiv HW(x) \pmod 2$. It is easy to see that for any two words x and y, we have $g(x \oplus y) = g(x) \oplus g(y)$, $g(x) \oplus g(x) = 0$, and $g(x \lll k) = g(x)$ for any $0 \leq k \leq 31$. Using these rules and the formulas in steps 5-7 above, one sees

$$g(V1[i] \oplus V2[j] \oplus V3[m]) \;=\; g(V1[i]) \oplus g(V2[j]) \oplus g(V3[m]) \;=$$

$$= \; g(T1) \oplus g(T2) \oplus g(T2) \oplus g(T3) \oplus g(0x8c591ca1) \oplus g(T3) \oplus g(T1) \oplus g(0xab8ec254) \;=$$

$$= \quad g(0x8c591ca1) \oplus g(0xab8ec254) \quad = \quad 0,$$

as asserted. □

In order to exploit Proposition 3, we first give a criterion that describes such a simultaneous update of the three words above in terms of the indices (i, j, m) in the current step, and the indices (i', j', m') in a certain step in the previous cycle of 64 steps. We denote the k-th step in the r-th cycle by S_k^r, and examine the triples of indices (i, j, m) corresponding to each step.

Definition 1. *We say Event E_k^r occurs if the following relations among indices are met during step S_k^r. Let $(T1, T2, T3) = (V1[a_1], V2[a_2], V3[a_3])$ be the words producing the output of this step (i.e., for this step, $(i, j, m) = (a_1, a_2, a_3)$; in particular, $a_1 = k$). The event occurs when these three conditions are satisfied:*

1. *In step $S_{a_3}^{r-1}$, the indices (i, j, m) are equal to (a_3, a_1, a_2).*
2. *In the steps S_i^{r-1} for all $i > a_3$, the indices satisfy $j \neq a_1$ and $m \neq a_2$.*
3. *In the steps S_i^r for all $i < a_1$, the indices satisfy $j \neq a_2$ and $m \neq a_3$.*

Note that if Event E_k^r occurs, then the values $T1$, $T2$, and $T3$ used for producing the key-stream word in step S_k^r are exactly of the form described in Proposition 3. Indeed, Condition 1 assures that $T1$, $T2$, and $T3$ were updated together in step $S_{a_3}^{r-1}$, and Conditions 2 and 3 assure that none of them was updated since the simultaneous update. In the next proposition we calculate the probability of occurrence of events E_k^r, based on our randomness assumption mentioned at the end of Section 2.

Proposition 4. *The probability of the event E_k^r is $Pr[E_k^r] = 2^{-13.20}(1 - 1/64)^{2k}$.*[4]

Proof. First, we calculate the probability of the event E_k^r if the triple of indices (i, j, m) used in step S_k^r is (a_1, a_2, a_3) (where $a_1 = k$). Since the indices j and m are each randomly distributed among 64 possible values, the probability that Condition 1 is satisfied is 2^{-12}. For the same reason, the probabilities that Conditions 2 and 3 are satisfied are $(1 - 1/64)^{2(63-a_3)}$, and

[4]Throughout this paper, numbers are rounded to the last decimal place.

$(1-1/64)^{2a_1}$, respectively. Since the three conditions are independent, the probability of the event E_k^r in this case is

$$Pr\Big[E_k^r|(a_1,a_2,a_3)\Big] = 2^{-12}(1-1/64)^{126-2a_3+2a_1}.$$

The unconditional probability of the event E_k^r is the average of the conditional probabilities over all the possible values of a_2 and a_3. Hence,

$$\begin{aligned} Pr[E_k^r] &= 2^{-6}\sum_{a_3=0}^{63} 2^{-12}(1-1/64)^{126-2a_3+2a_1} \\ &= 2^{-18}(1-1/64)^{126+2a_1}\sum_{a_3=0}^{63}(1-1/64)^{-2a_3} \approx 2^{-13.20}(1-1/64)^{2a_1}. \end{aligned}$$

The assertion follows since $a_1 = k$. □

Using Proposition 4 we can show that the parity of the Hamming weight of the key-stream words in SN3 is biased.

Proposition 5. *Consider a random key-stream word x output by the* SN3 *algorithm (at any stage of the key-stream generation), and denote by $g(x)$ the parity of the Hamming weight of x. Then $Pr[g(x)=0] = \frac{1}{2} + 2^{-15.40}$.*

Proof. Consider the key-stream word x_k at step S_k^r of the key-stream generation. By Proposition 4, the event E_k^r occurs with probability $Pr[E_k^r] = 2^{-13.2}(1-1/64)^{2k}$. By Proposition 3, if this event occurs, then $Pr[g(x_k) = 0] = 1$. On the other hand, Event E_k^r characterizes the conditions for all three values of T_1, T_2, T_3 to be simultaneously updated. If the event does not occur, some other random word(s) in the buffer influence the output, and by our randomness assumption we instead have $Pr[g(x_k)=0] = \frac{1}{2}$. Thus, in total,

$$\begin{aligned} Pr[g(x_k)=0] &= \Big(2^{-13.20}(1-1/64)^{2k}\Big)\cdot 1 + \Big(1-2^{-13.20}(1-1/64)^{2k}\Big)\cdot(1/2) \\ &= 1/2 + 2^{-14.20}(1-1/64)^{2k}. \end{aligned}$$

Finally, $Pr[g(x)=0]$ is the average over the possible values of k. Thus,

$$Pr[g(x)=0] = 1/2 + 2^{-6}\sum_{k=0}^{63} 2^{-14.20}(1-1/64)^{2k} = 1/2 + 2^{-15.40}.$$

This completes the proof of Proposition 5. □

Since for a random stream, the parity of the Hamming weights of the words is unbiased, we get the following:

Corollary 1. *The key-stream of* SN3 *can be distinguished from a random stream using less than* 2^{30} *key-stream words.*[5]

3.3 Experimental Results

Our computations above predict that $g(K_i)$ is zero slightly more often than it is 1, and in particular that $\sum_{i \le t} g(K_i)$ is roughly $\frac{t}{2} - 2^{-15.40}t$. (Were the cipher unbiased, the second term would be replaced by an error term of size roughly $O(t^{1/2})$.) We modified Maltchev's C code [11] to check this, and ran it on a Dell PowerEdge 2650 with an Intel Zeon 2.6GHZ processor and 2GB RAM. It took approximately 30 hours to generate each of 3 samples of 2^{40} output words. The empirical bias from our $3 \cdot 2^{40}$ trials was $-0.0000231 \approx 2^{-15.402}$, very close to our prediction. A plot of $\sum_{i \le t} g(K_i) - \frac{t}{2}$ from one of the trials is presented in Figure 1.

4 Distinguishing Attack on MV3

4.1 The Structure of MV3

The stream cipher MV3 [9] was designed by Keller *et al.* in 2006. The main goal of its design is to adapt the alleged RC4 structure to word-based stream ciphers, in which storing in memory a permutation of all the possible words is infeasible. The internal state of the cipher is an array of pseudo-random words, and the output rule is based on random accesses to the array determined by rapidly mixing random walks on directed graphs. In addition, the cipher uses a combination of different arithmetic word operations (XORs, additions, and multiplications), as well as cyclical rotations.

The structure of MV3 is the following: The heart of the internal state is an array T of 256 32-bit words, which is slowly updated during the key-stream

[5] We calculate the amount of key-stream required for a distinguishing attack using Theorem 1 of [16], which states that a key-stream length at least $c_0 q^{-2}$ is sufficient for getting an advantage of .5 over a random stream. Here q denotes the bias of the linear approximation, and $c_0 \approx .454936$ is the positive constant such that $\frac{1}{\sqrt{2\pi}} \int_{-\sqrt{c_0}}^{\sqrt{c_0}} \exp(-u^2/2)\, du = \frac{1}{2}$. (Please note that the value of c_0 was slightly misstated as $\approx .4624$ in [16].) For a detailed discussion, see [1].

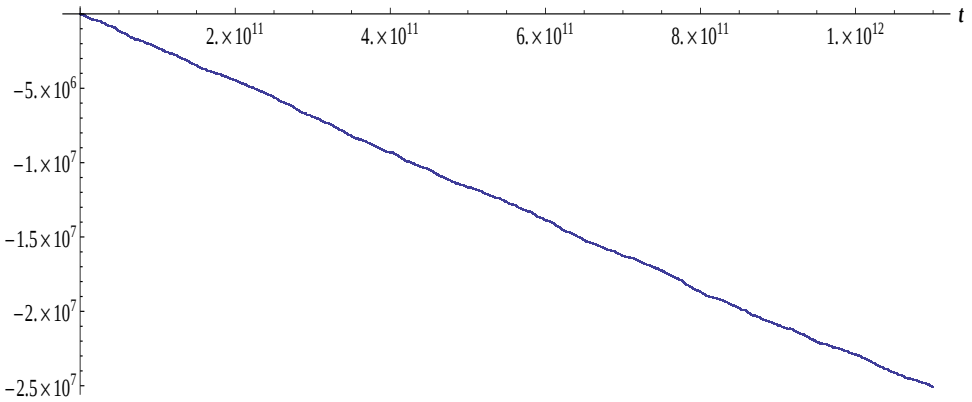

Figure 1: Cumulative sums $\sum_{i \le t} g(K_i) - \frac{t}{2}$ of the Hamming weight parity of SN3 output words. Were they indeed unbiased, the graph would be centered near zero, rather than having the negative slope it displays due to the bias calculated above.

generation phase. In addition, the internal state contains three buffers A, B, and C of 32 words each, that are cyclically shifted every 32 steps (namely, buffer A is discarded, buffer B becomes A, buffer C becomes B, and fresh values compose the new buffer C). The internal state is completed by several indices: secret word indices x and c and byte index j, that are updated using random walks on different directed graphs, and publicly known indices i (running from 0 to 31) and u (running from 0 to 255). The key-stream words are combinations of words taken from the A, B, and C buffers, where the array T is used for updating the indices and refilling the buffer C. The key-stream generation procedure is presented in the following pseudo-code:

Input: length *len*
Output: stream of length *len*
  repeat *len*/32 **times**
    for $i = 0$ **to** 31
      $j \leftarrow j + (B[i] \bmod 256)$
      $x \leftarrow x + T[j]$
      $C[i] \leftarrow (x \ggg 8)$
      output $(x \cdot c) \oplus A[9i + 5] \oplus (B[7i + 18] \ggg 16)$
    end for
    $u \leftarrow u + 1$

$T[u] \leftarrow T[u] + (T[j] \ggg 13)$
$c \leftarrow c + (A[0] \ggg 16)$
$c \leftarrow c \vee 1$
$c \leftarrow c^2$
$A \leftarrow B$, $B \leftarrow C$
end repeat

In the code, $+$ and $\cdot$ denote addition and multiplication modulo 2^{32}, respectively, and $(\ggg k)$ denotes cyclical rotation by k bits to the right. The symbol $\vee$ denotes a logical OR operation, and $\oplus$ denotes bitwise XOR.

We omit the key initialization phase of the cipher, since it does not affect our attacks. Instead, we assume that the internal state is initialized with perfectly random words. In particular, since the initial values of the indices x, c, and j are determined in the initialization phase, we assume that these values are random, in accordance with the working assumption at the end of Section 2.

4.2 Distinguishing Attack on a Simplified Variant of MV3

In order to explain the main idea of our attack clearly, we start with considering a simplified variant of the cipher in which some *cyclical rotations are removed*. Namely, the update rule of buffer C is replaced by the simpler rule

$$C[i] \leftarrow x, \tag{1}$$

and the output rule is replaced by

$$Out(i) = (x \cdot c) \oplus A[9i + 5] \oplus B[7i + 18].$$

The rest of the structure remains unchanged.

In this simplified variant, the attacker can concentrate on the least significant bits (LSBs) of the output words, for which modular additions are equivalent to bitwise XORs. Moreover, since the LSB of c is always equal to 1, the LSB of $x \cdot c$ is always equal to the LSB of x. Hence, if the attacker examines only the LSBs of the output words, the simplified variant of the cipher is equivalent to an even simpler variant in which the update rule of x is replaced by

$$x \leftarrow x \oplus T[j], \tag{2}$$

and the output rule is simply

$$Out(i) = C[i] \oplus A[9i + 5] \oplus B[7i + 18].$$

In the attack, we consider the XOR of two output words $Out(i_1)$ and $Out(i_2)$, given at steps i_1 and i_2 of the same 32-step loop in the key generation process, where $i_2 - i_1$ is even. We have

$$Out(i_1) \oplus Out(i_2) = \Big(C[i_1] \oplus C[i_2]\Big) \oplus \Big(A[9i_1+5] \oplus A[9i_2+5]\Big) \oplus \Big(B[7i_1+18] \oplus B[7i_2+18]\Big). \tag{3}$$

According to (1) and (2), the term $C[i_1] \oplus C[i_2]$ is the XOR of $(i_2 - i_1)$ randomly chosen elements of the array T and hence, by Proposition 2, its LSB is biased toward zero. Moreover, since the elements in buffers B and A are simply the elements of buffer C one or two loops before, the terms $A[9i_1 + 5] \oplus A[9i_2 + 5]$ and $B[7i_1 + 18] \oplus B[7i_2 + 18]$ are also XORs of an even number of randomly chosen elements of the array T, and thus, their LSBs are also biased toward zero. Because of our working assumption that the elements of A, B, and C are independent, these biases can be combined using Matsui's Piling-up Lemma [12] to compute the bias of the LSB of the expression (3). To be precise, denote by q_k the bias toward zero of the XOR of k random 0/1 elements (in the notation of Proposition 2, $q_k = p_k - 1/2$). Furthermore, let

$$k_1 = i_2 - i_1, \qquad k_2 = \Big|(9i_1 + 5) \bmod (32) - (9i_2 + 5) \bmod (32)\Big|,$$

$$k_3 = \Big|(7i_1 + 18) \bmod (32) - (7i_2 + 18) \bmod (32)\Big|.$$

Then by Matsui's Piling-up Lemma,

$$Pr\Big[\text{LSB}\Big(\Big(C[i_1] \oplus C[i_2]\Big) \oplus \Big(A[9i_1 + 5] \oplus A[9i_2 + 5]\Big) \oplus \Big(B[7i_1 + 18] \oplus B[7i_2 + 18]\Big)\Big) = 0\Big] = \frac{1}{2} + 4q_{k_1}q_{k_2}q_{k_3}.$$

An exhaustive check of the possible values of i_1 and i_2 (using the values of q_k computed in Proposition 2) shows that the right hand side assumes its maximal value when $i_2 - i_1 = 4$, for 24 of the 32 possible values of i_1. In these cases, $k_1 = k_2 = k_3 = 4$, and hence the right hand side equals

$$1/2 + 4\Big(\frac{3}{2 \cdot 256^2} - \frac{1}{256^3}\Big)^3 = 1/2 + 2^{-44.26}.$$

By Equation (3), this means that

$$Pr\Big[\text{LSB}(Out(i_1) \oplus Out(i_2)) = 0\Big] = 1/2 + 2^{-44.26}. \tag{4}$$

Since for a random string, the expression in the left hand side is unbiased, Formula (4) can be used to distinguish the output of this simplified variant of MV3 from a random string using less than 2^{88} words of output.

4.3 Distinguishing Attack on MV3

When attacking the original version of MV3, the attacker cannot restrict himself to the LSBs of the words in buffer C. Due to the cyclical rotations, even if the attacker considers only the LSBs of the output words, he has to deal also with bits 8 and 24 of buffer C that influence the output through buffers A and B. As a result, the update rule $x \leftarrow x + T[j]$ is no longer equivalent to the simpler $x \leftarrow x \oplus T[j]$, and thus the attacker has to overcome the composition of XORs and modular additions. In particular, the observation used in Section 4.2 is not valid anymore: while the XOR of an even number of random elements of the T array is biased in each of its bits, the same is not true for their *modular sum*.[6]

In order to overcome these difficulties, we consider the XOR of four output words in the same 32-step loop, having the following special structure:

$$(Out(i_1) \oplus Out(i_1 + l)) \oplus (Out(i_2) \oplus Out(i_2 + l)),$$

for an appropriate choice of i_1, i_2, and l. We note that while the simplified attack presented in Section 4.2 resembles the differential cryptanalysis technique [3], the attack presented here is an instance of higher-order differential cryptanalysis [10]. We have:

$$\Big(Out(i_1) \oplus Out(i_1 + l)\Big) \oplus \Big(Out(i_2) \oplus Out(i_2 + l)\Big) = \tag{5}$$

$$= \Big(((C[i_1] \lll 8) \cdot c) \oplus ((C[i_1 + l] \lll 8) \cdot c)\Big) \oplus \Big(((C[i_2] \lll 8) \cdot c) \oplus ((C[i_2 + l] \lll 8) \cdot c)\Big) \oplus \tag{6}$$

[6] For example, consider the sum $T[j_1] + T[j_2]$ for random j_1 and j_2. If $j_1 \neq j_2$ then clearly the sum is unbiased. If $j_1 = j_2$ then the sum $T[j_1] + T[j_2]$ is simply a left shift of $T[j_1]$ by one bit. The LSB of this value is necessarily zero, but any other bit k is unbiased since it is equal to bit $k - 1$ of $T[j_1]$. Hence, in total, bit k of $T[j_1] + T[j_2]$ is unbiased for all $k > 0$.

$$\oplus \Big(A[9i_1+5] \oplus A[9(i_1+l)+5]\Big) \oplus \Big(A[9i_2+5] \oplus A[9(i_2+l)+5]\Big) \oplus \quad (7)$$

$$\Big((B[7i_1{+}18] \ggg 16) \oplus (B[7(i_1{+}l){+}18] \ggg 16)\Big) \oplus \Big((B[7i_2{+}18] \ggg 16) \oplus (B[7(i_2{+}l){+}18] \ggg 16)\Big). \quad (8)$$

Like in the simplified attack, we consider each of the terms (6), (7), and (8) separately, and show that a certain linear combination of its bits is biased toward zero. Then, we use Matsui's Piling-up Lemma [12] to combine these biases into a bias of the entire expression.

For the sake of simplicity, we consider first a simplified version of the term (6) in which the cyclical rotations and the multiplication with c are removed. We shall show later that both the rotations and the multiplication by c do not change the conclusions of our argument. Hence, we consider the simplified term:

$$(C[i_1] \oplus C[i_1+l]) \oplus (C[i_2] \oplus C[i_2+l]). \quad (9)$$

The basic observation we use in our attack is the following:

Observation 1. *Recall that the elements of the buffer C are determined by consecutive values of x. By the update rule of x, if $i_1 < (i_1+l) \bmod 32$, then the relation between $C[i_1]$ and $C[i_1+l]$ is (ignoring the cyclical rotation):*

$$C[i_1+l] = C[i_1] + \Big(T[j(i_1+1)] + T[j(i_1+2)] + \ldots + T[j(i_1+l)]\Big),$$

where $j(i)$ denotes the value of the index j at step i of the loop. That is, $C[i_1+l]$ is obtained from $C[i_1]$ by adding a sum of l random elements of the T array. Similarly, if $i_2 < (i_2+l) \bmod 32$, then $C[i_2+l]$ is obtained from $C[i_2]$ by adding a sum of l random elements of the T array.[7] If the indices of the two sequences of random elements are equal (possibly in a different order), then we have

$$C[i_1+l] - C[i_1] = C[i_2+l] - C[i_2], \quad (10)$$

[7] If both $i_1 < (i_1+l) \bmod 32$ and $i_2 < (i_2+l) \bmod 32$ do not hold, then the argument presented below can still be applied, when (i_1, i_2) is replaced by $(i_1+l, i_2+l) \bmod 32$, and l is replaced by $32-l$. If only one of the two conditions holds, then the sequences considered in the sequel are of different lengths, and hence our attack cannot be applied. Hence, the attacker has to choose values (i_1, i_2, l) such that either none or two of the conditions $i_1 < (i_1+l) \bmod 32$ and $i_2 < (i_2+l) \bmod 32$ are satisfied.

where "−" denotes subtraction (mod 2^{32}). We show that the relation (10) can be used to detect a set of biases satisfied by Expression (9), which is invariant under cyclical rotations.

In order to exploit expression (10) to detect a bias in the expression (9), we have to find a relation between the subtraction operation used in (10) and the XOR operation used in (9). Note that we seek for a relation that is invariant under cyclical rotations, and hence the obvious relations using the MSB or the LSB are not sufficient for our purpose. We use the following result, due to Cho and Pieprzyk [5]:

Proposition 6. *([5, Corollary 2]) Let Γ_i denote the 32-bit linear mask in which only bits i and $i+1$ are non-zero. Thus, for any 32-bit word x, we have $\Gamma_i(x) = x_i \oplus x_{i+1}$. Let x, y, and K be random 32-bit words. Then for all $0 \le i \le 30$,*

$$Pr\Big[\Gamma_i\left((x+K) \oplus (y+K) \oplus x \oplus y\right)\Big] \;=\; \frac{2}{3} + \frac{2^{-2i-2}}{3} \;>\; \frac{2}{3}\,.$$

Using Proposition 6, we can give a precise form to Observation 1.

Proposition 7. *Consider $0 \le i_1 < i_2 \le 31$ and $1 \le l \le 31$, such that the two conditions $i_1 < (i_1 + l) \bmod 32$ and $i_2 < (i_2 + l) \bmod 32$ are satisfied. Let $l_1 = \min(l, i_2 - i_1)$. For any $0 \le i \le 30$,*

$$Pr\Big[\Gamma_i\Big(((C[i_1] \lll 8)\cdot c) \oplus ((C[i_1{+}l] \lll 8)\cdot c) \oplus ((C[i_2] \lll 8)\cdot c) \oplus ((C[i_2{+}l] \lll 8)\cdot c)\Big) = 0\Big]$$
$$> \;\frac{1}{2} + \frac{256!\, l_1!}{6\,(256 - l_1)!\, 256^{2l_1}}.$$

Proof. We consider first a simplified version of the statement, without the cyclical rotation and multiplication by c. As noted in Observation 1, we have

$$C[i_1 + l] \;=\; C[i_1] \;+\; \Big(T[j(i_1+1)] + T[j(i_1+2)] + \cdots + T[j(i_1+l)]\Big),$$

and similarly with i_1 replaced by i_2. Denote by E_1 the following event: the two multisets $\{j(i_1{+}1), j(i_1{+}2), \ldots, j(i_1{+}l)\}$ and $\{j(i_2{+}1), j(i_2{+}2), \ldots, j(i_2{+}l)\}$ are equal. It is clear that if the event E_1 occurs, then

$$T[j(i_1{+}1)]{+}T[j(i_1{+}2)]{+}\cdots{+}T[j(i_1{+}l)] = T[j(i_2{+}1)]{+}T[j(i_2{+}2)]{+}\cdots{+}T[j(i_2{+}l)]. \tag{11}$$

Thus, we can apply Proposition 6 with $x = C[i_1]$, $y = C[i_2]$, and $K = T[j(i_1+1)] + T[j(i_1+2)] + \cdots + T[j(i_1+l)]$, to get

$$Pr\left[\,\Gamma_i\left(C[i_1] \oplus C[i_1+l] \oplus C[i_2] \oplus C[i_2+l]\right) = 0 \,\middle|\, E_1\right] \;>\; \tfrac{2}{3}.$$

If the event E_1 does not occur, then by the randomness of the elements of the T array, the expression in the left hand side is unbiased. Hence, in total,

$$Pr\left[\,\Gamma_i\left(C[i_1] \oplus C[i_1+l] \oplus C[i_2] \oplus C[i_2+l]\right) = 0\,\right] \;>\; \tfrac{1}{2} \;+\; \tfrac{Pr[E_1]}{6}\,. \qquad (12)$$

The probability of the event E_1 depends on the *overlapping* between the sequences $(i_1+1, i_1+2, \cdots, i_1+l)$ and $(i_2+1, i_2+2, \cdots, i_2+l)$. If some of the indices overlap, then the corresponding $T[j(i)]$ terms can be removed from Equation 11, thus simplifying the equation. It is easy to see that the number of non-overlapping indices in the sequences is $l_1 = \min(l, i_2 - i_1)$.[8] Hence, Equation 11 is reduced to

$$T[j(i_1{+}1)]{+}\cdots{+}T[j(i_1{+}l_1)] \;=\; T[j(i_2{+}(l{-}l_1{+}1))]{+}\cdots{+}T[j(i_2{+}l)]. \quad (13)$$

Furthermore, since in Equation (13) the indices of i are distinct, we can use the randomness assumption on the index j, to assume that the indices $j(\cdot)$ in both sides of the equation are independently uniformly distributed in $\{0, 1, \ldots, 255\}$. The probability of E_1 is bounded from below by the probability of the following event E_2: all the elements of each of the multisets $\{j(i_1+1), j(i_1+2), \ldots, j(i_1+l_1)\}$ and $\{j(i_2+(l-l_1+1)), j(i_2+(l-l_1+2)), \ldots, j(i_2+l)\}$ are distinct, and the two multisets are equal. It is easy to see that

$$Pr[E_2] \;=\; \frac{256 \cdot 255 \cdot \ldots \cdot (256-l_1+1)}{256^{l_1}} \frac{l_1!}{256^{l_1}} \;=\; \frac{256!\, l_1!}{(256-l_1)!\, 256^{2l_1}}.$$

Substituting this into (12) proves the proposition when the cyclical rotations and multiplications by c are omitted. However, it is clear from the proof that if one rotates all words in the buffer C by a constant number of bits, and multiplies them all by the same word c, a similar argument applies. □

[8] For example, the equation $T[j(1)] + T[j(2)] + \cdots + T[j(9)] = T[j(4)] + T[j(5)] + \cdots + T[j(12)]$ is equivalent to the simpler equation $T[j(1)] + T[j(2)] + T[j(3)] = T[j(10)] + T[j(11)] + T[j(12)]$..

Now consider Equation (5) at the beginning of this subsection. Proposition 7 asserts that the term (6) is biased toward zero. Since the elements of buffers A and B are obtained from elements of buffer C in previous loops, the argument of Proposition 7 is valid also for the terms (7) and (8), and hence both of them are also biased toward zero. The exact lower bounds on the biases are obtained by substituting the triples $(9i_1 + 5, 9i_2 + 5, 9l) \bmod (32)$ and $(7i_1 + 18, 7i_2 + 18, 7l) \bmod (32)$ in Proposition 7 in the place of (i_1, i_2, l). In total, Proposition 7 can be applied to all the three terms (6), (7), and (8) if the following three pairs of conditions are satisfied:[9]

$$i_1 < (i_1 + l) \bmod (32), \qquad i_2 < (i_2 + l) \bmod (32),$$

$$9i_1{+}5 \bmod (32) < 9(i_1{+}l){+}5 \bmod (32), \qquad 9i_2{+}5 \bmod (32) < 9(i_2{+}l){+}5 \bmod (32),$$

$$7i_1{+}18 \bmod (32) < 7(i_1{+}l){+}18 \bmod (32), \qquad 7i_2{+}18 \bmod (32) < 7(i_2{+}l){+}18 \bmod (32). \tag{14}$$

For terms (7) and (8), we replace the value l_1 computed in Proposition 7 for the term (6) by

$$l_2 = \min(9l \bmod (32),\ |(9i_1 + 5) \bmod (32) - (9i_2 + 5) \bmod (32)|) \text{ and}$$

$$l_3 = \min(7l \bmod (32),\ |(7i_1 + 18) \bmod (32) - (7i_2 + 18) \bmod (32)|),$$

respectively. Combining the biases using Matsui's Piling-up Lemma [12], we thus get:

Corollary 2. *Let (i_1, i_2, l) be such that $0 \le i_1 < i_2 \le 31$, $1 \le l \le 31$, and the six conditions described in (14) are satisfied. Then for all $0 \le i \le 30$ one has*

$$Pr\left[\Gamma_i \left(Out(i_1) \oplus Out(i_1 + 4) \oplus Out(i_2) \oplus Out(i_2 + 4)\right) = 0\right] > \tfrac{1}{2} + 4\, r(l_1)\, r(l_2)\, r(l_3),$$

where l_1, l_2, and l_3 are as defined above, and $r(l_i) = \frac{256!\, l_i!}{6\,(256-l_i)!\, 256^{2l_i}}$.

An exhaustive check of all the possible triples (i_1, i_2, l) (performed by a computer program) shows that the right hand side assumes its maximal value for the quadruple $(0, 1, 14, 15)$ as well as 21 more quadruples $(i_1, i_1{+}l, i_2, i_2{+}l)$

[9] As noted before, each pair of conditions can be replaced by the complement conditions, by slightly modifying the argument. This change can be done for each of the pairs independently, and hence, in total there are eight sets of six conditions for which the attack can be applied.

listed in the footnote below.[10] For each of these quadruples, we have $l_1 = 1$ and $l_2 = l_3 = 2$, and thus,

$$Pr\left[\Gamma_i \left(Out(i_1) \oplus Out(i_1+4) \oplus Out(i_2) \oplus Out(i_2+4)\right) = 0\right] \; > \; \tfrac{1}{2} + 2^{-43.77}. \tag{15}$$

Finally, we note that in the attack we can use 31 linear approximations in parallel, each for a different choice of i in the linear mask Γ_i. Hence, in total, in each 32-step loop the attacker can use 22 quadruples of $(i_1, i_1 + l, i_2, i_2 + l)$ and 31 different approximations for each of them.[11] Since the bias of each approximation is $2^{-43.77}$, the data complexity of the distinguisher is $(0.454936 \cdot 2^{87.53} \cdot 32)/(22 \cdot 31) = 2^{81.99}$ words of key-stream. Summarizing the attack, we have:

Corollary 3. *The key-stream of* MV3 *can be distinguished from a random string using less then* 2^{82} *words of key-stream.*

5 Summary and Conclusions

In this paper we examined a structural distinguishing attack introduced in [16], applicable to stream ciphers whose internal state consists of arrays of pseudo-random words, and whose output is a relatively simple function of the internal state. We presented a rigorous proof of the heuristic assumptions used in the attack in basic cases, along with experimental evidence for the precision of the assumptions in more complicated cases. In addition, we extended the technique, allowing it to handle more complicated update and output rules, and used it to mount a distinguishing attack on the cipher MV3 [9], requiring less than 2^{82} words of output.

[10] The values of the 22 optimal quadruples $(i_1, i_1 + l, i_2, i_2 + l)$ are:

$$(0,1,14,15),(0,1,18,19),(1,2,15,16),(3,4,21,22),(4,5,18,19),(4,5,22,23),$$
$$(5,6,19,20),(5,6,23,24),(6,7,20,21),(6,7,24,25),(7,8,21,22),(7,8,25,26),$$
$$(8,9,22,23),(8,9,26,27),(9,10,23,24),(11,12,29,30),(12,13,26,27),(12,13,30,31),$$
$$(13,14,27,28),(14,15,28,29),(15,16,29,30),(16,17,30,31).$$

[11] For a precise treatment of linear attacks using a combination of multiple linear approximations, see [4].

Our attacks, along with previously published distinguishing attacks on stream ciphers based on arrays (e.g., [16, 19]), show that unless the output rule is very complex, it is difficult to avoid "academic" distinguishing attacks. However, perhaps this should not necessarily be considered a weakness of such ciphers. Instead, since extremely long key-streams are not used in real-life applications, it seems reasonable to ask what complexity of a distinguishing attack is reasonable to tolerate before deeming a stream cipher insecure with respect to it.[12]

6 Acknowledgements

We would like to thank Orr Dunkelman, Ilya Mironov, and Ramarathnam Venkatesan for fruitful discussions, Alex Sherman for helping us to conduct the experiments on SN3, Souradyuti Paul for discussions on the constant in Theorem 1 of [16], and Fahimeh Mohebipoor and Mohammad Ali Orumiehchi for discussing with us the initial ideas of the attack on MV3.

References

[1] C. Baigneres, P. Junod, and S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, Advances in Cryptology - proceedings of Asiacrypt 2004, Lecture Notes in Computer Science **3329**, pp. 432–450, Springer-Verlag, 2004.

[2] Eli Biham and Jennifer Seberry, *Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays*, eStream, ECRYPT Stream Cipher Project, Report 2005/023, 2005.

[3] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[4] Alex Biryukov, Christophe De Canniere, and Michael Quisquater, *On Multiple Linear Approximations*, Advances in Cryptology - proceedings

[12] We note that such statements were made in the specifications of the stream ciphers Py [2] and HC-256 [19]. In both cases, the designers were aware of the existence of distinguishing attacks faster than exhaustive search, but did not consider them a threat to the security of the cipher due to their very large data requirements.

of CRYPTO 2004, Lecture Notes in Computer Science **3152**, pp. 1–22, Springer-Verlag, 2004.

[5] Joo Yeon Cho and Joseph Pieprzyk, *Multiple Modular Additions and Crossword Puzzle Attack on NLSv2*, proceedings of ISC 2007, Lecture Notes in Computer Science **4779**, pp. 230–248, Springer-Verlag, 2007.

[6] William Feller, *An Introduction to Probability Theory and its Applications*, 3rd ed., John Wiley & Sons Inc., 1968.

[7] S. Fluhrer and D. McGrew, *Statistical Analysis of the Alleged RC4 Keystream Generator*, proceedings of FSE 2000, Lecture Notes in Computer Science **1978**, pp. 19–30, Springer-Verlag, 2001.

[8] C. Gong, K.C. Gupta, M. Hell, and Y. Nawaz, *Towards a General RC4-like Keystream Generator*, proceedings of CISC 2005, Lecture Notes in Computer Science **3822**, pp. 162–174, Springer-Verlag, 2005.

[9] Nathan Keller, Stephen D. Miller, Ilya Mironov, and Ramarathnam Venkatesan, *MV3: A New Word Based Stream Cipher Using Rapid Mixing and Revolving Buffers*, Topics in Cryptology - proceedings of CT-RSA 2007, Lecture Notes in Computer Science **4377**, pp. 1–19, Springer-Verlag, 2006.

[10] Lars Knudsen, *Truncated and Higher-Order Differentials*, proceedings of FSE 1994, Lecture Notes in Computer Science **1008**, pp 196–211, Springer-Verlag, 1995.

[11] Simeon V. Maltchev, *The SN3 Stream Cipher*, Available on-line at: `http://www.geocities.com/smaltchev/sn3.html`.

[12] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology — proceedings of Eurocrypt 1993, Lecture Notes in Computer Science **765**, pp. 386–397, Springer-Verlag, 1994.

[13] Y. Nawaz, K.C. Gupta, and C. Gong, *A 32-bit RC4-like Keystream Generator*, Cryptology ePrint Archive, 2005/175.

[14] M. A. Orumiehchi and S. F. Mohebipoor, *Distinguishing Attack on SN3 Stream Cipher*, proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1392–1395, 2008.

[15] Mohammad Ali Orumiehchi, S. Fahimeh Mohebbipoor, and Hossein Ghodosi, *Cryptanalysis of MV3 Stream Cipher*, proceedings of CANS 2008, Lecture Notes in Computer Science **5339**, pp. 240–251, Springer-Verlag, 2008.

[16] Souradyuti Paul and Bart Preneel, *On the (In)security of Stream Ciphers Based on Arrays and Modular Addition*, Advances in Cryptology - proceedings of ASIACRYPT 2006, Lecture Notes in Computer Science **4284**, pp. 69–83, Springer-Verlag, 2006.

[17] Souradyuti Paul, *Cryptanalysis of Stream Ciphers Based on Arrays and Modular Addition*, Ph.D. Thesis, Katholieke Universiteit Leuven, 2006.

[18] G. Rose and P. Hawkes, *On the Applicability of Distinguishing Attacks Against Stream Ciphers*, preproceedings of the 3rd NESSIE workshop, available online at: `http://eprint.iacr.org/2002/142.pdf`.

[19] Hongjun Wu, *A New Stream Cipher HC-256*, proceedings of Fast Software Encryption 2004, Lecture Notes in Computer Science **3017**, pp. 226–244, Springer-Verlag, 2004.

A Appendix

In this appendix we examine the attacks on SN3 and MV3 presented in [14] and [15], respectively, and show that their correct data complexities are 2^{38} key-stream words for SN3 (instead of $2^{28.2}$ claimed by the authors), and 2^{82} key-stream words for MV3 (instead of 2^{63} claimed by the authors).

A.1 The Attack on SN3 Presented in [14]

The attack uses some of the ideas used in our attack on SN3, along with different techniques. Unfortunately, their work suffers from two computational flaws:

1. On Page 3, immediately above Figure 2, it is asserted that

$$1 \times \frac{1}{64} \times \frac{1}{64} \times 1 \times \frac{1}{64} \quad = \quad 2^{-15}.$$

Replacing the right hand side by the correct value of 2^{-18} results in reducing the probability of occurrence of Event I (computed at the bottom of Page 3) from $2^{-4}(1 + 2^{3.9} \times 2^{-15})$ to $2^{-4}(1 + 2^{3.9} \times 2^{-18})$.

2. Immediately following this, the authors assert (without proof) that since the probability of Event I for a random permutation is 2^{-4}, the cipher can therefore be distinguished with $2^{22.2}$ cycles of 64 output words. This includes the implicit claim that two Binomial random variables $Bin(n, p)$ with means $2^{-4}(1 + 2^{3.9} \times 2^{-15})$ and 2^{-4} can be distinguished with $2^{22.2}$ samples. This claim is incorrect. Since the standard deviation of the variables is $\sqrt{np(1-p)}$ with $p \approx 2^{-4}$, the number of required samples must satisfy $\sqrt{np(1-p)} \approx 2^{-15}n$, which leads to $n \approx 2^{26}$.

Combining the two corrections, we get that the probability of Event I is $2^{-4}(1 + 2^{3.9} \times 2^{-18})$, and hence the number of samples required for the distinguishing is approximately 2^{32}. Since each sample is obtained from a cycle of 64 words, the overall data complexity of the attack is close to 2^{38} key-stream words.

A.2 The Attack on MV3 Presented in [15]

The attack is essentially the same as the attack presented in our paper, but claims a different overall complexity due to the following errors:

1. On Page 244 it is stated that

$$\Pr[\text{Const1} = \text{Const2}] \quad = \quad \frac{2}{\binom{256}{2}} \quad \approx \quad 2^{-13.99}.$$

This statement is incorrect. We have Const1 = Const2 if two unordered pairs of integers between 1 and 256 are equal. The probability that two such ordered pairs are equal is 2^{-16}. Since the order does not matter, the probability is approximately doubled. Hence, the correct probability is approximately 2^{-15}.

2. Formula (5) on Page 244 states that

$$\Pr([x_t \cdot c]_{0,1} \oplus [(x_t + \text{Const1}) \cdot c]_{0,1} \oplus [x_\tau \cdot c]_{0,1} \oplus \\ [(x_\tau + \text{Const2}) \cdot c]_{0,1} = 0) \approx 1/2 + 0.66 \cdot 2^{-13.99}.$$

The right hand side is incorrect. The formula follows from combining the relation Const1 = Const2 which holds (by the author's claims) with probability $2^{-13.99}$, and the approximation of the relation between XOR and addition (taken from the paper of Cho and Pieprzyk) which holds with probability 2/3. Hence, the correct probability is

$$2^{-13.99} \cdot (2/3) + (1 - 2^{-13.99}) \cdot (1/2) = 1/2 + 2^{-13.99} \cdot (1/6).$$

3. The bias of expression (6) is calculated on page 245 to be

$$\frac{1}{2}(1 + 2^2 \cdot (\frac{2}{3})^3 2^{-8} (2^{-13.99})^2) \quad \approx \quad \frac{1}{2} + 2^{-36.73}.$$

This formula combines the biases of three expressions of the form of (5) using Matsui's Piling-up Lemma [12]. The lemma states that if the biases are q_1, q_2, q_3, then the overall bias is $2^2 q_1 q_2 q_3$. In our case, two of the biases are equal to $(1/6) \cdot 2^{-15}$ (instead of $(2/3) \cdot 2^{-13.99}$, as we explained above). For similar reasons, the third bias is equal to $(1/6) \cdot 2^{-8}$ (instead of $(2/3) \cdot 2^{-8}$). Hence, the total probability should be updated to

$$\frac{1}{2} + 2^2 \cdot (\frac{1}{6})^3 \, 2^{-8} \, (2^{-15})^2) \quad \approx \quad \frac{1}{2} + 2^{-43.73}.$$

Note that this probability agrees with the probability computed in our attack in (15).

4. The number of samples required for the distinguishing attack is computed as

$$n = 2^{-9.53} \times 2^{72.46} = 2^{62.93},$$

due to the parallel application of multiple approximations. However, this number counts 32-word cycles and not single key-stream words (only a full cycle contains $2^{9.53}$ approximations that can be used in parallel). Hence, the overall data complexity in key-stream words should be multiplied by 2^5.

Combining all the corrections, the bias of the approximation should be multiplied by 2^7, and the final data complexity should be multiplied by 2^5. Hence, the correct data complexity of the attack is

$$2^{62.93} \cdot (2^7)^2 \cdot 2^5 = 2^{81.93},$$

consistent with our attack.